Apache Spark Word Count App

User

2025-01-13, 01:15:56

Generate a Apache Spark application using Scala language to solve the problem! Given a plain text UTF-8 file URL and a output CSV file URL, write in the output file the occurrence sum of each word in the input file.

Assistant Phi 3 mini Instruct

2025-01-13, 01:15:56

Here's a Scala-based Apache Spark application that reads a plain text UTF-8 file from a given URL, computes the occurrence sum of each word, and writes the results to a CSV file at a specified output URL. ```scala import org.apache.spark.sql.SparkSession import org.apache.spark.sql.functions._ object WordCountApp { def main(args: Array[String]): Unit = { val inputFileUrl = args(0) // The URL of the input text file val outputFileUrl = args(1) // The URL where the output CSV file will be saved val spark = SparkSession.builder .appName("WordCountApp") .getOrCreate() // Read the text file into a DataFrame val lines = spark.readStream .option("encoding", "UTF-8") .option("multiline", "true") .text(inputFileUrl) // Split the lines into words val words = lines.as[String].flatMap(_.split("\\s+")) // Generate running word count val wordCounts = words.groupBy("value").count() // Write the word counts to the output CSV file wordCounts.writeStream .outputMode("complete") .format("csv") .option("header", "true") .option("path", outputFileUrl) .option("checkpointLocation", "/path/to/checkpoint/directory") .start() spark.sparkContext.getConf.get("spark.sql.shuffle.partitions", "10") spark.streams.awaitAnyTermination() } } ``` To run this application, you need to provide the input file URL and output file URL as command-line arguments. For example: ```scala WordCountApp.scala http://example.com/input.txt http://example.com/output.csv ``` Please note that you need to replace `/path/to/checkpoint/directory` with an actual directory path where Spark Streaming checkpoints will be stored. This application uses Spark's structured streaming API to process the text file in real-time. The word counts are computed using a `groupBy` operation on the `value` column (which represents each word) and then counting the occurrences. The results are written to a CSV file in "complete" output mode, which means that for each batch of data, all the records will be emitted. Make sure to have the necessary dependencies and configurations set up for your Spark environment before running this application.<|end|>